Nicht für den Prüfling bestimmt

I Erläuterungen

Voraussetzungen gemäß KCBG und Abiturerlassen BG jeweils in der für den Abiturjahrgang geltenden Fassung

Standardbezug
Die nachfolgend ausgewiesenen Kompetenzbereiche sind für die Bearbeitung der jeweiligen Aufgabe besonders bedeutsam. Darüber hinaus können weitere, hier nicht ausgewiesene Kompetenzbereiche für die Bearbeitung der Aufgabe nachrangig bedeutsam sein, zumal die Kompetenzbereiche in engem Bezug zueinander stehen. Die Operationalisierung des Bezugs zu den Kompetenzbereichen des Standardbezugs erfolgt in Abschnitt II.

Aufgabe	**Kompetenzbereiche**				
	K1	**K2**	**K3**	**K4**	**K5**
1.1			X		
1.2			X		
1.3				X	
2.1				X	
2.2				X	
2.3	X				
2.4	X				
2.5	X				
2.6					X
3.1		X			
3.2			X		

Inhaltlicher Bezug
Die nachfolgend ausgewiesenen Themenfelder sind die wesentliche inhaltliche Grundlage für die vorliegenden Aufgaben. Darüber hinaus können weitere, hier nicht explizit ausgewiesene Themenfelder für die Bearbeitung nachrangig bedeutsam sein.
Q1: Objektorientierte Softwareentwicklung
Q2: Digitale Steuerungstechnik
Q3: Prozessautomatisierung
verbindliche Themenfelder: Objektmodellierung (Q1.1), Implementierung von Klassen und ihren Beziehungen (Q1.2), Synthese statischer und sequentieller Logikschaltungen (Q2.1), Mikrocontroller (Q2.2), Einführung in die Prozessautomatisierung (Q3.1), Server-Client-Kommunikation (Q3.2)

II Lösungshinweise

In den nachfolgenden Lösungshinweisen sind alle wesentlichen Gesichtspunkte, die bei der Bearbeitung der einzelnen Aufgaben zu berücksichtigen sind, konkret genannt und diejenigen Lösungswege aufgezeigt, welche die Prüflinge erfahrungsgemäß einschlagen werden. Selbstverständlich sind jedoch Lösungswege, die von den vorgegebenen abweichen, aber als gleichwertig betrachtet werden können, ebenso zu akzeptieren.

Nicht für den Prüfling bestimmt

Aufg.	erwartete Leistungen	BE I	BE II	BE III
1.1	entwickeln, bestimmen Servicetechniker — Datenzentrale einrichten Datenzentrale einrichten <<include>> anmelden anmelden <<include>> User eingeben anmelden <<include>> Passwort eingeben Datenzentrale einrichten <<include>> Geräteadresse eingeben Datenzentrale einrichten <<include>> Gerätetyp eingeben Geräteprotokoll drucken <<extend>> Datenzentrale einrichten Gerät kalibrieren <<extend>> Datenzentrale einrichten Crewmitglied — Bildschirmgerät konfigurieren Bildschirm in Fenster unterteilen <<extend>> Bildschirmgerät konfigurieren Zuordnung der Daten zu den Fenstern <<extend>> Bildschirmgerät konfigurieren Warnschwellen setzen <<extend>> Bildschirmgerät konfigurieren Hinweis: Eine Vererbung bei „Bildschirmgerät konfigurieren“ ist gleichwertig zu bewerten. entwickeln		3	3
	bestimmen	6		
1.2	modellieren, darstellen **Datenzentrale**: - username : String; - passwort : String; + login() : void; + geraetHinzufuegen() : void; + geraetLoeschen() : void Datenzentrale 1 ↔ * Geraet <<abstract>> ***Geraet***: - geraeteAdresse : int; - geraeteTyp : String **Windsensor** (erbt von Geraet): - windgeschwindigkeit : int; - windrichtung : int; + messen() : void **Positionsleuchte** (erbt von Geraet): - farbe : String; - eingeschaltet : boolean; + einschalten() : void; + ausschalten() : void **Steuerradmotor** (erbt von Geraet): - steuersignal : int; - modusAuto : boolean; + autopilotEin() : void; + autopilotAus() : void **Navigation** (erbt von Geraet): - sollkurs : double; - istkurs : double; - steuersignal : int; + steuersignalErmitteln() : void Hinweis: Die grau hinterlegte Klasse Navigation ist nicht verlangt. modellieren		3	3
	darstellen	7		

1.3	implementieren
	
	`public class TestUI {`
	`  private static Navigation navigation;`
	`  public static void main(String args[]){`
	`    navigation = new Navigation();`
	`    navigation.zyklus();`
			6	9

Aufg.	erwartete Leistungen	BE		
		I	II	III

	<pre> navigation.zyklus();
	    navigation.zyklus();
	    }
	}
	
	import java.util.ArrayList;
	public class Navigation {
	  private double istKurs;
	  private double sollKurs;
	  private ArrayList<Position> positionsListe;
	  private int lfdNr = -1;
	  private GPS myGPS;
	  public Navigation() {
	    this.positionsListe = new ArrayList<Position>();
	    this.myGPS = new GPS();
	  }
	  public void positionHinzufuegen(Position position)
	{
	    positionsListe.add(position);
	    lfdNr++;
	  }
	  public void zyklus() {
	    positionHinzufuegen(myGPS.positionAktualisieren());
	    if(lfdNr!=0){
	      istKurs = istKursErmitteln(positionsListe.get(lfdNr),
	                positionsListe.get(lfdNr-1));
	    System.out.println("aktueller Kurs = "+istKurs+
	     " , Element in der positionsListe = "+lfdNr);
	  }
	  public double istKursErmitteln(Position
	          positionAktuell, Position positionAlt) {
	    return 0;
	  }
	}
	
	public class Position {
	  private int laengengrad;
	  private int breitengrad;
	  public Position() {
	  }
	}
	
	public class GPS {
	  private Position position;
	  public GPS() {
	    position = new Position();
	  }
	  public Position positionAktualisieren(){
	    return (position);
	  }
	}</pre>
	Hinweis: Der grau hinterlegte Teil der Lösung ist nicht zu bewerten.			

	Summe 40	**13**	**12**	**15**

Aufg.	erwartete Leistungen	BE I	BE II	BE III
2.1	entwickeln, zeichnen			

```
setup:
Port B <- output
Port D <- input

loop:
  Port B, RUECK <- 1
  time10ms
  Port B, RUECK <- 0
  Port B, START <- 1
  time10ms
  Port B, START <- 0
  time4sek
  Port B, STOPP <- 1
  time10ms
  Port B, STOPP <- 0
  zaehler <- IN Port D
  zaehler , Bit 1 == 1 ?
    ja:   tmp <- 1
    nein: tmp <- 0
  zaehler durch 4 teilen
  R1 <- zaehler + tmp
  R2, Bit 0 == 1 ?
    ja:   Port B, ANKERLICHT <- 1
    nein: Port B, ANKERLICHT <- 0
  time26sek
endlos
```

Aufg.	erwartete Leistungen	BE I	BE II	BE III
	entwickeln		5	
	zeichnen	5		

Aufg.	erwartete Leistungen	BE		
		I	II	III
2.2	implementieren		3	9

```
.equ ANKERLICHT   = 0
.equ START        = 1
.equ STOPP        = 2
.equ RUECK        = 3
.def tmp          = r16
.def zaehler      = r13


setup:
ldi  tmp, 0b00001111 // setze Eingänge und Ausgänge
out  DDRB, tmp       // im Data Direction Register B
ldi  tmp,0b00000000 // setze Eingänge und Ausgänge
out  DDRD, tmp      // im Data Direction Register B

loop:
sbi  PORTB, RUECK   // 10ms Impuls Rücksetzen des Zählers
call time10ms
cbi  PORTB, RUECK
sbi  PORTB, START   // 10ms Impuls Starten des Zählers
call time10ms
cbi  PORTB, START
call time4s         // 4 s Messzeit
sbi  PORTB, STOPP   // 10ms Impuls Stoppen des Zählers
call time10ms
cbi  PORTB, STOPP
in   zaehler, PIND  // Laden des Zählerstandes
clr  tmp            // tmp für das Runden löschen
sbrs zaehler, 1     // Ist das Bit 1 im zaehler gesetzt,
rjmp runden         // wenn ja tmp auf 1 für das Runden
ldi  tmp, 1
runden:
lsr  zaehler        // zaehler durch 4 teilen
lsr  zaehler
add  zaehler, tmp   // 0 oder 1 für das Runden
hinzuaddieren
mov  r1, zaehler    // für die Kommunikationssoftware
verfügbar
sbrs r2, 0          // Ist das Bit 0 im r2 gesetzt
rjmp ankerLichtaus
sbi PORTB, ANKERLICHT // Ankerlicht an
rjmp wait26s
ankerLichtaus:
cbi PORTB, Ankerlicht // Ankerlicht aus
wait26s:
call time26s
rjmp loop
```

Aufg.	**erwartete Leistungen**	**BE**		
		I	**II**	**III**
2.3	entwickeln, zeichnen Mikro-controller — Start → S (RS - FF); Stopp → R (RS - FF); Q → count (Zähler); windSpeed → Zähler; Rueck → rueck (Zähler); Zaehlerstand: Zähler → Mikrocontroller			
	entwickeln		3	
	zeichnen	3		
2.4	erläutern Während der Initialisierung wird ein Merkerbit in einem Register auf 0 gesetzt. Wird in der Endlosschleife festgestellt, dass der Eingang windSpeed auf high geht, wird das Merkerbit auf 1 gesetzt. Wenn bei einem späteren Durchgang durch die Endlosschleife windSpeed auf low geht und das Merkerbit gesetzt ist, dann wird der Zähler um eins erhöht und das Merkerbit zurückgesetzt.	2	3	
2.5	nennen, erläutern Dieser Signalverlauf kann z. B. beim Betätigen von Tastern auftreten. Das sogenannte „Prellen“ tritt durch das Nachschwingen der Kontaktzungen innerhalb des Tasters auf, wenn er betätigt wird. Das Nachschwingen sorgt dafür, dass der Taster scheinbar mehrfach betätigt wird. Hier kann nach dem Detektieren des High-Signals auf der Signal-Leitung eine Warteschleife gestartet werden, die mindestens T_max und kleiner als T_min dauert. Anschließend kann der Zähler erhöht werden.			
	nennen	3		
	erläutern		2	
2.6	analysieren, diskutieren Die binär codierte Scheibe hat das niederwertigste Bit im äußeren Ring. Schwarz bedeutet eine 1 und Weiß (transparent) eine 0. Von binär 0 auf 1 wechselt nur in einem Ring der Zustand, nämlich im Äußeren. Von 1 auf 2 wechseln in zwei Ringen die Zustände, der Äußere von 1 auf 0 und der Mittlere von 0 auf 1. Beim Gray Code wechselt bei jedem Übergang nur in einem Ring der Zustand. Da die Codescheiben von IR-Lichtschranken abgetastet werden, kann es bei binär-codierten Scheiben zu Mischdaten kommen, d. h., dass beim Übergang von 1 auf 2 kurzzeitig beide äußeren Ringe eine 1 liefern, was einer 3 und nicht einer 2 entspräche.			
	analysieren		2	1
	diskutieren			1
	Summe 42	**13**	**18**	**11**

Aufg.	erwartete Leistungen	BE I	BE II	BE III
3.1	analysieren, beschreiben Die Datenzentrale ermöglicht den Aufbau einer Client-Server-Verbindung durch Instanziierung eines `ServerSockets`, hier mit dem Namen `serverSocket`. Dieses Objekt `serverSocket` horcht auf einem Port und erwartet dort Anfragen von Clients. In der Aufgabe ist der Client das Objekt `wind` der Klasse `WindSensor`. Das Objekt `wind` instanziiert ein Objekt `clientSocket` der Klasse `Socket` und ruft in diesem die Operation `connect()` auf. Diese Operation fragt bei dem Serverrechner auf dem Port, auf dem der Server horcht, eine Verbindung an. Wird diese akzeptiert, instanziiert `serverSocket` ein Objekt `clientSide` der Klasse `Socket`. Nun besteht die Client-Server-Verbindung zwischen den Objekten `clientSide` und `clientSocket`, was durch den Rückgabewert `true` der Operation `connect()` dem Objekt `wind` mitgeteilt wird. `wind` geht anschließend in die Operation `holeAnfrage()`, in der das Protokoll, welches in dem RFC-Dokument beschrieben wird, abgearbeitet wird. Die Anfragen werden vom Server initiiert und in der Aufgabe möchte der Server zunächst die Windgeschwindigkeit ermitteln. Hierzu sendet er unter Verwendung der Operation `write()` den Code `sendWindSpeed`. Dieses wird in der Operation `holeAnfrage()` im Objekt `wind` clientseitig in einer Abfrageschleife erkannt. Laut RFC-Protokoll muss der Client hierauf mit dem Integerwert der Umdrehungen pro Sekunde antworten. Im Beispiel sind das 14. Serverseitig ist im Protokoll das Einlesen des Integerwertes vorgesehen. Als nächstes möchte der Server den Status des Ankerlichtes mittels des Codes `setAnkerLicht` übermitteln. Hierzu schreibt der Server direkt im Anschluss den Status des Ankerlichtes mittels `write()`, im Beispiel eine 0. Der Client liest den Wert mittels `read()` und bestätigt den Empfang mit dem Senden mittels `write()` eines Acknowledge `ack`. Nach diesem Schema können weitere Anfragen bearbeitet werden, bis das Schließen der Verbindung erfolgt. Dieses wird von beiden Seiten mit dem Aufruf der Operationen `close()` an die Objekte `clientSocket` und `clientSide` ausgelöst. Hieran schließt sich auch das Beenden des Objektes `serverSocket` an, ebenfalls mittels der Operation `close()` in diesem Objekt.			
	analysieren		6	
	beschreiben	4		
3.2	entwickeln holeAnfrage(): eingabe <- clientSocket.readline() eingabe == "sendWindSpeed" ? (ja / nein) ja: clientSocket.write(umdrehungen) — nein: ∅ eingabe == "setAnkerLicht" ? (ja / nein) ja: status <- clientSocket.readline() statusAnkerlicht <- status clientSocket.write("ack") — nein: ∅ eingabe == "getErrorCode" ? (ja / nein) ja: clientSocket.write(errorCode) — nein: ∅		4	4
	Summe 18	**4**	**10**	**4**

III Bewertung und Beurteilung

Die Bewertung und Beurteilung erfolgt unter Beachtung der nachfolgenden Vorgaben nach § 33 der Oberstufen- und Abiturverordnung (OAVO) in der jeweils geltenden Fassung. Bei der Bewertung und Beurteilung der sprachlichen Richtigkeit in der deutschen Sprache sind die Bestimmungen des § 9 Abs. 12 Satz 3 OAVO in Verbindung mit Anlage 9b anzuwenden.

Bei der Bewertung und Beurteilung der Übersetzungsleistung in den Fächern Latein und Altgriechisch sind die Bestimmungen des § 9 Abs. 14 OAVO in Verbindung mit Anlage 9c anzuwenden.

Der Fehlerindex ist nach Anlage 9b zu § 9 Abs. 12 OAVO zu berechnen. Für die Ermittlung der Punkte nach Anlage 9a zu § 9 Abs. 12 OAVO sowie Anlage 9c zu § 9 Abs. 14 OAVO wird jeweils der ganzzahlige nicht gerundete Prozentsatz bzw. Fehlerindex zugrunde gelegt.

Für die Bewertung in den modernen Fremdsprachen ist der „Erlass zur Bewertung und Beurteilung von schriftlichen Arbeiten in allen Grund- und Leistungskursen der neu beginnenden und fortgeführten modernen Fremdsprachen in der gymnasialen Oberstufe, dem beruflichen Gymnasium, dem Abendgymnasium und dem Hessenkolleg“ vom 7. August 2020 (ABl. S. 519) zugrunde zu legen. Demnach erfolgt die Bewertung und Beurteilung mit der Maßgabe, dass lediglich bei der Ermittlung des Prüfungsergebnisses (Note) aus Prüfungsteil 1 und 2 gerundet wird.

Darüber hinaus sind die Vorgaben der Erlasse „Hinweise zur Vorbereitung auf die schriftlichen Abiturprüfungen (Abiturerlass)“, „Hinweise zur Vorbereitung auf die schriftlichen Abiturprüfungen im beruflichen Gymnasium (fachrichtungs-/ schwerpunktbezogene Fächer) (Abiturerlass BG)“ und „Durchführungsbestimmungen zum Landesabitur“ in der für den Abiturjahrgang geltenden Fassung zu beachten.

Als Kriterien für die Bewertung und Beurteilung dienen unter Beachtung der Zielsetzung der gymnasialen Oberstufe nach § 1 Abs. 2 OAVO neben dem Inhaltlichen auch die in den Kerncurricula genannten überfachlichen Kompetenzen, insbesondere die Sprachkompetenz und Wissenschaftspropädeutik; dies zeigt sich u.a. in qualitativen Merkmalen wie Strukturierung, Differenziertheit, (fach-)sprachlicher Gestaltung und Schlüssigkeit der Argumentation.

Im Fach Technische Informatik besteht die Prüfungsleistung aus der Bearbeitung eines Vorschlags, wofür insgesamt maximal 100 BE vergeben werden können. Ein Prüfungsergebnis von **5 Punkten (ausreichend)** setzt voraus, dass mindestens 45% der zu vergebenden BE erreicht werden. Ein Prüfungsergebnis von **11 Punkten (gut)** setzt voraus, dass mindestens 75% der zu vergebenden BE erreicht werden.

Gewichtung der Aufgaben und Zuordnung der Bewertungseinheiten zu den Anforderungsbereichen

Aufgabe	Bewertungseinheiten in den Anforderungsbereichen			Summe
	AFB I	AFB II	AFB III	
1	13	12	15	**40**
2	13	18	11	**42**
3	4	10	4	**18**
Summe	**30**	**40**	**30**	**100**

Die auf die Anforderungsbereiche verteilten Bewertungseinheiten innerhalb der Aufgaben sind als Richtwerte zu verstehen.